Data Stream Clustering with Affinity Propagation

Xiangliang Zhang, Cyril Furtlehner, Cecile Germain-Renaud, Michèle Sebag

Data Stream Clustering with Affinity Propagation

Xiangliang Zhang*, Cyril Furtlehner†, Cécile Germain-Renaud†, Michèle Sebag†
*King Abdullah University of Science and Technology (KAUST), Saudi Arabia
Xiangliang.Zhang@kaust.edu.sa
†TAO – INRIA, CNRS, University of Paris-Sud 11, Orsay, France
{Cyril.Furtlehner, Cecile.Germain, Michele.Sebag}@lri.fr

Abstract—Data stream clustering provides insights into the underlying patterns of data flows. This paper focuses on selecting the best representatives from clusters of streaming data. There are two main challenges: how to cluster with the best representatives and how to handle the evolving patterns that are important characteristics of streaming data with dynamic distributions. We employ the Affinity Propagation (AP) algorithm presented in 2007 by Frey and Dueck for the first challenge, as it offers good guarantees of clustering optimality for selecting exemplars. The second challenging problem is solved by change detection. The presented STRAP algorithm combines AP with a statistical change point detection test; the clustering model is rebuilt whenever the test detects a change in the underlying data distribution. Besides the validation on two benchmark data sets, the presented algorithm is validated on a real-world application, monitoring the data flow of jobs submitted to the EGEE grid.

Index Terms—Streaming Data Clustering, Affinity Propagation, Grid Monitoring, Autonomic Computing

1 INTRODUCTION

Dealing with non-stationary distributions is a key issue for many application domains, e.g., sensor network or traffic data monitoring [1]. At the crossroad of databases, data mining and machine learning, Data Streaming is the discipline specifically concerned with handling large-scale datasets in an online fashion [2], [3]; many data streaming algorithms have been adapted from clustering algorithms, e.g., the partitioning method k-means [4], [5], [6], [7], [8] or the density-based method *DbScan* [9], [10].

This paper focuses on learning a generative model of a data stream, with some specific features:

- the generative model is expressed through a set of exemplars, i.e., actual data items as opposed to centroids in order to accommodate complex (e.g., relational) domains;
- the generative model is available at any time, for the sake of monitoring applications;
- the changes in the underlying distribution are detected through statistical hypothesis testing.

These specificities are motivated by the targeted domain of application, Autonomic Computing [11], rooted in the fact that the ever increasing complexity of computational systems calls for new approaches to system management [12], enforcing *self-modeling, self-configuring, self-healing and self-optimizing* facilities [13]. The vision behind Autonomic Computing is to provide a computational system with an analog of the biological immune system, seamlessly enforcing quite a few vital activities (e.g., breathing and heart-beating) according to the demands of the external environment and the internal state of the organism.

The presented work specifically concerns a large-scale grid system, the EGEE/EGI Grid, which is the largest e-science grid infrastructure worldwide. Created through a series of EU projects[1], it involves over 400,000 cores and 200 Petabytes storage. It supports more than 1 million jobs per day on a 24/24, 7/7 basis. The applicative goal of the presented approach is to process the log files that describe the flow of jobs submitted to and processed by gLite, the major EGEE/EGI middleware, and to provide the system administrator with a dashboard of the job stream.

The proposed approaches proceed by extending the Affinity Propagation (AP) algorithm, a message passing-based clustering method proposed by Frey and Dueck [14]. Formulating the clustering problem in terms of energy minimization, AP outputs a set of clusters, each of which is characterized by an actual data item, referred to as an *exemplar*; the penalty value parameter controls the cost of adding another exemplar. AP provides some asymptotic guarantees of optimality of the solution. The price to pay for these properties is AP's quadratic computational complexity, forbidding its usage on large scale datasets.

The online version of AP, called STRAP[2], was firstly

1. *Enabling Grid for E-SciencE* and *European Grid Infrastructure*, http://www.egi.eu

2. The programming codes of STRAP are available to download at http://mine.kaust.edu.sa/Pages/Software.aspx.

proposed and studied in our previous work [15], [16]. The proposed algorithm proceeds by incrementally updating the current model if the current data item fits the model, and putting it in a *reservoir* otherwise. A Change Point Detection (CPD) test, the Page-Hinkley test [17], [18], detects the changes of distribution by monitoring the proportion of data items sent to the reservoir (the proportion of *outliers*). Upon triggering the CPD test, a new model is rebuilt from the current model and the data items in the reservoir.

Extending our previous work [15], [16], this paper presents a complete STRAP algorithm for data stream clustering with comprehensive analysis in theoretical and empirical manners. The improvements over our previous work are summarized as follows. First, this paper investigates a real-time adapted CPD test, rather than using empirically setting or model-based optimization in CPD test [15], [16]. The CPD test plays an important role in STRAP for catching the evolving distribution. The adaptive threshold can achieve clusters in better quality, cause less outliers and require less computing time.

Second, the performance of STRAP algorithm for extracting representative exemplars is theoretically analyzed. STRAP extracts exemplars from streaming data for building a summary model. CPD test enables STRAP to catch drifting exemplars that significantly deviate away. A minor drifting exemplar modifies the synopsis of its associated cluster but cannot replace its associated exemplar. For the first time, we theoretically analyze the upper bound of the distortion loss caused by minor drifting exemplars.

Third, we study the complexity and memory usage of STRAP, which are important efficiency factors for streaming algorithms. Our analysis shows that the memory usage of STRAP mainly depends on the number of exemplars and outliers, which is small and varies a little in the streaming process. The time complexity of STRAP is quadratic w.r.t. the number of exemplars and outliers. Experimental evaluation confirms our analysis regarding the efficiency of STRAP.

Last but not least, extensive sensitivity analysis and experimental evaluation are conducted. We employed the KDD'99 intrusion detection benchmark data, comparatively to the *DenStream* algorithm [9], a recent URL stream data set [19] and the EGEE job stream. The new experimental results validate the effectiveness of the proposed method.

This paper is organized as follows. Section 2 briefly reviews related work. Section 3 presents Affinity Propagation for the sake of completeness and describes the proposed Weighted AP. Section 4 describes STRAP, extending AP to data streaming. Section 5 reports the experimental validation on KDD'99 data, a URLs stream and a real-world application, monitoring the EGEE job data flows. The paper concludes with perspectives for further research in Section 6.

2 STATE OF THE ART

Data streaming, one of the major Data Mining tasks [20], [21], faces two additional difficulties compared to traditional data clustering problem. Both are related to the non-stationarity of the data distribution. On one hand, functionally, a streaming algorithm must maintain an efficient trade-off between noise filtering (outliers must be discarded) and model updating (most generally, upon detecting a change in the data distribution, the model must be appropriately updated). On the other hand at a more general level, a streaming algorithm must adjust its own parameters, e.g., the number k of clusters can hardly be determined beforehand.

One-scan Divide-and-Conquer approaches have been widely used to cluster data streams, e.g., extending k-means [22] or k-median [4], [5] approaches. The basic idea is to segment the data stream and process each subset in turn, which might prevent the algorithm from catching the distribution changes in a timely manner; likewise, it adversely affects the adjustment of the number k of clusters. In [23], data samples flowing in are categorized as *discardable* (outliers), or *compressible* (accounted for by the current model), or *to be retained* in the RAM buffer. Clustering, e.g., k-means, is iteratively applied, considering the sufficient statistics of compressed and discarded points, and the retained points in RAM.

Recently, Ackermann et al. proposed an online k-means by maintaining a small sketch of the input using the merge-and-reduce technique [7]. The proposed StreamKM++ algorithm can obtain better clustering results (minimizing the sum of squared errors) than BIRCH [24] but takes more computing time than BIRCH. Shindler et al. improved the streaming approximation for Euclidean k-means where k is not known as input and data points are sequentially read to form clusters [8]. Their approach can provide better approximation guarantee and is efficient with complexity $o(nk)$. Both of these two recent algorithms aim at efficiently grouping data into clusters with high quality. Efficiency in computing time and memory usage is achieved by sequentially loading large-scale data as streams. When the algorithms terminate, they produce clusters of the large data set, instead of forming clusters at any time step when data flow in.

As emphasized by Dong et al. [25], a core issue in data streaming is to detect the changes in the data flow distribution. A two-level scheme first proposed by Aggarwal et al. [6] proceeds as follows: the evolving data stream is processed and summarized online; these summaries are grouped into clusters offline, using k-means [6] or density-based clustering methods [9], [10]. The latter approaches can reveal clusters of arbitrary shapes while k-means only reveal spherical clusters. These two approaches are respectively parameterized from the number k of clusters,

and the density threshold; whereas one might argue that the number of clusters is more brittle than the density threshold in the non-stationary case, it is most generally desirable to adjust the parameters online.

Let us detail the *DenStream* algorithm, which upgrades the density-based clustering algorithm *DbScan* to data streaming along the lines of the two-level scheme [9]. Within the online level, micro-clusters are built online to reflect the stream, creating a new one iff new data items arrive and fall outside of the scope of the nearest micro-cluster; otherwise, the new data item is merged to the nearest micro-cluster. In the former case, an outlier micro-cluster is created; for the sake of the memory management, an (old) micro-cluster is deleted or two micro-clusters are merged. *DenStream* maintains a weight attached to each micro-cluster, which is used to decide whether it should be deleted upon building an outlier micro-cluster, or upgrading an outlier micro-cluster into a potential one. While the online micro-clusters reflect the underlying probability density function, the actual clusters are only built upon the user's query, using a variant of *DbScan* on the micro-clusters.

Based on the online maintenance and offline clustering strategy, Dai et al. also proposed a Clustering on Demand framework [26]. In the online phase, wavelet and regression analyses were used to construct summary hierarchies. In the offline phase, approximations of desired substreams are retrieved from summary hierarchies according to clustering queries. Like *DenStream*, the clustering model is available upon request.

It must be emphasized that both Divide-and-Conquer and two-level schemes keep their computational load within reasonable limits as they only build the data model upon the user's explicit request. In the rest of time, they only maintain a summary of the data, which makes them ill-suited to applications such as system monitoring where the data model has to be continuously available at all times.

TABLE 1
Notations in algorithms

Notation	Description
$\mathcal{X}$=$\{x_1, ..., x_N\}$	data items to be clustered
K	the number of clusters
$x_{i_1}, \ldots, x_{i_K}$	K exemplars in $\mathcal{X}$
$\mathbf{c}$=$(c_1, ..., c_N)$	mapping between items and exemplars
$E[\mathbf{c}]$	objective function of clustering
$S(x_i, x_j)$	similarity of x_i and x_j
$d(x_i, x_j)$	distance of x_i and x_j
σ	preference penalty, defined by $S(x_i, x_i)$
ϵ_i	average mutual distance of aggregated items to x_i
$(e_i, n_i, \Sigma_i, t_i)$	STRAP model with e_i–exemplar, n_i–the number of items associated to e_i, Σ_i–the distortion of e_i, and t_i–the last time stamp when an item was associated to e_i
T	the number of items for initialization
ε	fit threshold in STRAP
Δ	the length of sliding window
$p_t, \bar{p}_t, m_t, M_t$	statistical variables in Page-Hinkley test
λ	threshold for triggering change detection
u_t	deviation of outlier x_t from STRAP model
f	λ-factor, the number of witnesses of changes

3 (Weighted) Affinity Propagation

For the sake of self-containedness, this section first describes the Affinity Propagation (AP) algorithm. An extension, Weighted AP, is then presented to deal with duplicated and generally weighted data items. The notations used in this paper are presented in Table 1.

3.1 Affinity Propagation

Let $\mathcal{X} = \{x_1, \ldots, x_N\}$ be a set of items, and K denote a positive integer. The k-medoids problem aims at finding K items in $\mathcal{X}$, referred to as exemplars and denoted as $x_{i_1}, \ldots, x_{i_K}$, such that they minimize the sum, over all items x_j, of the minimal squared distance between x_j and $x_{i_k}, k = 1 \ldots K$. The Affinity Propagation approach proposes an equivalent formalization of the k-medoids problem, defined in terms of energy minimization. It solves the optimization problem

$$\mathbf{c}^* = \text{argmin}\big(E[\mathbf{c}]\big), \tag{1}$$

with

$$E[\mathbf{c}] = -\sum_{i=1}^{N} S(x_i, x_{c_i}) - \sum_{i=1}^{N} \ln \chi_i^{(p)}[\mathbf{c}] \tag{2}$$

where $\mathbf{c} = (c_1, \ldots, c_N)$ is the mapping between data and exemplars, $S(x_i, x_{c_i})$ is the similarity between x_i and its exemplar $x_{c_i} \in \mathcal{X}$, set to negative squared distance $-d(x_i, x_{c_i})^2$ if $i \neq c_i$. A free parameter called the *preference penalty* is the cost incurred for being oneself exemplar:

$$\sigma \stackrel{\text{def}}{=} S(x_i, x_i), \qquad \forall i, \tag{3}$$

$\chi_i^{(p)}[\mathbf{c}]$ is a set of constraints controlling the clustering structure. $\ln \chi_i^{(p)}[\mathbf{c}] \to -\infty$ if $x_{c_i} \neq x_i$ and $\exists j \; x_{c_j} = x_i$, which implies that if x_i is selected as an exemplar by some items, it has to be its own exemplar. Otherwise, $\ln \chi_i^{(p)}[\mathbf{c}] = 0$. The energy function thus enforces a tradeoff between the distortion, i.e., the sum over all items of the squared error $d(x_i, x_{c_i})^2$, and the cost of the model, that is $\sigma \times |\mathbf{c}|$ if $|\mathbf{c}|$ denotes the number of exemplars retained. Equation (2) thus does not directly specify the number of exemplars to be found, as opposed to k-medoids. Instead, the number of exemplars in the solution depends on penalty σ; note that $\sigma = 0$ yields a trivial solution, selecting every item as an exemplar.

A message passing algorithm is employed to solve the optimization problem defined by Equation (2), considering two types of messages: availability messages $a(i, k)$ express the accumulated evidence for x_k to be selected as the best exemplar for x_i; responsibility messages $r(i, k)$ express the fact that x_k is suitable to be the exemplar of x_i.

3.2 Weighted AP

The first extension, called Weighted AP (WAP), is proposed to deal with multiply-defined items and dense aggregations of items. Let dataset $\mathcal{E}' = \{(x_i, n_i)\}$ involve n_i copies of item x_i or n_i items aggregated into a single item x_i with small average mutual distance ϵ_i, for $i = 1 \ldots L$. Let us consider the similarity matrix S' defined as:

$$S'(x_i, x_j) = \begin{cases} n_i S(x_i, x_j) & \text{if } i \neq j \\ \sigma + (n_i - 1) \times \epsilon_i & \text{otherwise, with } \epsilon_i \geq 0 \end{cases}$$

Proposition 3.1: The combinatorial optimization problem of finding $\mathbf{c} : \{1 \ldots L\}$ minimizing

$$E'[\mathbf{c}] = -\sum_{i=1}^{L} S'(x_i, x_{c_i}) - \sum_{\mu=1}^{L} \ln \chi_{\mu}^{(p)}[\mathbf{c}] \qquad (4)$$

is equivalent, for $\epsilon_i = 0$, to the optimization problem defined by Equation (2) for $\mathcal{E}$ made of the union of n_i copies of x_i, for $i = 1 \ldots L$.

Proof: In the optimization problem defined by Equation (2), assume that x_i actually represents a set of n_i identical copies; the penalty $S(x_i, x_j)$ of selecting x_j as exemplar of x_i is the cost of selecting x_j as exemplar for each one of these copies. Therefore $S'(x_i, x_j) = n_i \times S(x_i, x_j)$.
Likewise, let x_i be unfolded as a set of n_i (almost) identical copies $\{x_{i_1}, \ldots, x_{i_{n_i}}\}$, and let us assume that one of them, say x_{i_1} is selected as exemplar. One thus pays the penalty σ, plus the sum of the dissimilarities between x_{i_1} and the other copies in x_i, modeled as $(n_i - 1)\epsilon_i$. Constant ϵ_i thus models the average dissimilarity among the n_i copies of x_i. □

4 STRAP: EXTENDING AP TO DATA STREAMING

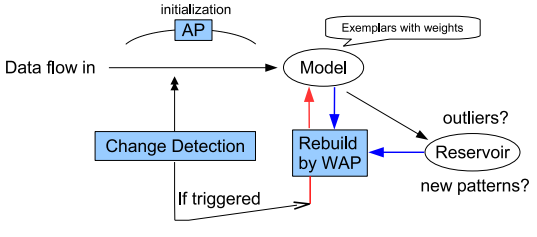


Fig. 1. Diagram of STRAP algorithm

This section aims at extending AP to data streaming, specifically achieving online clustering in the case of non-stationary data distributions. The resulting algorithm, called STRAP, involves four main steps (Algorithm 1 with a diagram in Figure 1):

1) The first bunch of data is used by AP to identify the first exemplars and initialize the stream model.
2) As the stream flows in, each data item x_t is compared to the exemplars; if too far from the nearest exemplar, x_t is put in the reservoir, otherwise the stream model is updated accordingly (Section 4.1).
3) The data distribution is checked for change point detection, using a statistical test, the Page-Hinkley (Section 4.2).
4) Upon triggering the change detection test, or if the number of outliers exceeds the reservoir size, the stream model is rebuilt based on the current model and reservoir, using WAP (Section 4.3).

Algorithm 1 STRAP Algorithm

```
Data stream x_1, ... x_t, ...; fit threshold ε
Init
  AP(x_1, ..., x_T) → STRAP Model                Section 4.1
  Reservoir = {}
for t > T do
  Compute e_i = nearest exemplar to x_t          Section 4.1
  if d(x_t, e_i) < ε then
    Update STRAP model                           Section 4.1
  else
    Reservoir ← x_t
  end if
                                                 Section 4.2
  if Restart criterion then
    Rebuild STRAP model                          Section 4.3
    Reservoir = {}
  end if
end for
```

Contrasting with the state of the art (Section 2), STRAP builds a model of the data flow which is available at any time step. Its performance will be empirically assessed in Section 5.

4.1 AP-based Model and Update

The model of the data stream used in STRAP is inspired from *DbScan* [27] and *DenStream* [9]. It consists of a set of 4-tuple $(e_i, n_i, \Sigma_i, t_i)$, where e_i ranges over the exemplars, n_i is the number of items associated to exemplar e_i, Σ_i is the distortion of e_i (sum of $d(x, e_i)^2$, where x ranges over all items associated to e_i), and t_i is the last time stamp when an item was associated to e_i.

At time t, item x_t is considered and its nearest (w.r.t. distance d) exemplar e_i in the current model is selected; if $d(x_t, e_i)$ is less than some threshold ε, heuristically set to the average distance between points and exemplars in the initial model, x_t is affected to the i-th cluster and the model is updated accordingly; otherwise, x_t is considered to be an outlier, and put into the reservoir.

In order to prevent the number of exemplars from growing beyond control, one must be able to forget the exemplars that have not been visited for a long time, as they may describe an obsolete model. Accordingly, a user-specified window length Δ is introduced;

when item x_t is associated to exemplar x_i, the model update is thus defined as:

$$\begin{array}{l} n_i := n_i \times \left(\frac{\Delta}{\Delta+(t-t_i)} + \frac{1}{n_i+1} \right) \\ \Sigma_i := \Sigma_i \times \frac{\Delta}{\Delta+(t-t_i)} + \frac{n_i}{n_i+1} d(x_t, x_i)^2 \\ t_i := t \end{array} \quad (5)$$

Calculations show that the above update rules enforce the model stability if exemplar x_i is selected on average by n_i examples during the last Δ time steps. The sensitivity analysis w.r.t. Δ is discussed in Section 5.3.

4.2 Restart Criterion

A key difficulty in Data Streaming is to detect a change in the generative process underlying the data stream, referred to as *drift*, which requires the stream model to be updated as soon as possible. The main difficulty lies in the fact that the first data samples reflecting the new components can hardly be distinguished from outliers. Masud et al. proposed to detect the concept drift as appearance of novel classes [28]. In their work, supervised methods were used to determine the decision boundary of training data. Instances lying outside the decision boundary and with strong cohesion among each other were reported as a novel class drifting from the current model.

In continuous settings, the model update is most usually dealt with by gradually moving the cluster centers [27]. In discrete settings, (part of) the cluster centers must be recomputed and a new optimization phase must be launched. A restart criterion is thus needed to trigger the stream model update.

The simplest option is based on the number of data items in the reservoir; when it exceeds the reservoir size, the restart criterion is automatically triggered.

A more sophisticated restart criterion[3] is based on statistical change point detection (CPD) tests. While quite a few multi-variate CPD tests have been proposed, in particular to deal with structured data (see e.g. [29], [30] and references therein), a computationally frugal test is needed in the data streaming context since it is run at every time step. For this reason, a scalar CPD test was chosen, specifically the Page-Hinkley (PH) change point detection test [17], [18]. Let us briefly remind the basics of PH test, before describing how it is integrated in STRAP framework.

4.2.1 The Page-Hinkley test

Let p_u denote the realization of the observed random variable p at time u. Denote $\bar{p}_t$ (respectively m_t) the empirical average of the p_u (respectively the sum of the differences between p_u and $\bar{p}_u$) on the time interval $[1, t]$:

$$\begin{array}{c} \bar{p}_t = \frac{1}{t} \sum_{u=1}^{t} p_u \\ m_t = \sum_{u=1}^{t} (p_u - \bar{p}_u + \delta) \end{array} \quad (6)$$

The PH test proceeds by maintaining the maximum value M_t of the m_u ($M_t = max\{m_1, \dots m_t\}$), and triggering the test if the gap between M_t and m_t is above a threshold λ, parameterizing the PH test,

$$\text{PH triggered iff } PH_t = M_t - m_t > \lambda$$

The PH test is theoretically grounded for testing a negative jump in the mean of a Gaussian distribution (a symmetric form exists for a positive jump). More precisely, p_u is assumed to be a Gaussian variable with mean μ_0 before the change point, and μ_1 after, with $\mu_1 < \mu_0$. PH_t is essentially a running (up to time t) estimator of the log-likelihood of distributions without and with jump. δ is a positive real-value that controls the test model: it should be half the size of the jump when μ_0 and μ_1 are known; if unknown, it is usually set to a small value, e.g., half the minimum jump considered significant model wise.

To get an intuition of the PH test, let us consider a simplistic example, where the value of the random variable is always equal to its mean: up to time t_0, $p_u = \mu_0$, and after t_0, $p_u = \mu_1$.
For $t \leq t_0$, $m_t = t\delta$, thus $M_t = m_t$ and $PH_t = 0$.
For $t > t_0$, $m_t = t_0\delta + \sum_{u=t_0+1}^{t} (\mu_1 - \bar{p}_u + \delta)$. Shortly after the change, the influence of new values on the empirical average is limited, thus $\bar{p}_u \approx \mu_0$, $m_t < M_{t_0}$,

$$PH_t \approx (t - t_0)(\mu_0 - \mu_1 - \delta) \quad (7)$$

PH_t is positive and quasi linearly increases w.r.t. the number of time steps.

The parameter λ is usually set after observing p_u for some time and depends on the desired alarm rate. Its adaptive adjustment is described in next section.

4.2.2 The test variables

As aforementioned, scalar CPD tests are preferred for the sake of computational efficiency, which requires scalar variables to be defined and to be able to reflect the state of the streaming process. In our earlier work, the scalar variable was set to the fraction of outliers observed in a sliding time window [16]. A new approach is presented in this paper.

Each data item x_t is associated to its nearest exemplar e. In the case where x_t is an outlier ($d(x_t, e) \geq \varepsilon$), x_t is marked to visit e at time step t. We note $t_1 < t_2 \dots < t_j$ the time steps of the consecutive outliers, and $u_t = d(x_t, e)$.

The general goal is to detect two distinct modalities of the apparition of new clusters. One is spatial: outliers begin coalescing in a region of space with small volume compared to the closest cluster; in this case, the variance of u_{t_j} will decrease. The second

3. In the following, the most new outlier replaces the oldest one in case the number of outliers exceeds the reservoir size. The total number of outliers since the current model was built is maintained and will serve to build the next model.

is temporal: the intensity of the stochastic process counting the outliers increases; this may for instance account for the case where widely separated clusters of equal spatial volume appear.

It then comes naturally to associate to each outlier the random variable defined as an inverse intensity-weighted empirical variance of the distance. For sake of memory, and also in order to forget stale information, this variable is computed on a sliding window of size l:

$$p_{t_l} = \sqrt{\frac{1}{l}\sum_{i=1}^{l}(1+\log(t_i - t_{i-1}))(u_{t_i} - \frac{1}{l}\sum_{k=1}^{l} u_{t_k})^2}$$

In the following, we note p_t instead of p_{t_l} for simplicity. A decreasing of p_t suggests outliers are both frequent and dense in a given region of the item space.

The drift/change in p_t is monitored throughout the Page-Hinkley test. A heuristic justification for this is possible when the intensity is constant and the u_t is normally distributed. Then, the empirical variance follows a chi-squared distribution with l degrees of liberty before the change point, and a translated $\chi^2(l)$ after. As the square root of a chi-squared distribution converges fast to a normal distribution, testing for a negative jump (decrease) in mean is reasonable.

The next question is how to calibrate the threshold parameter λ, which controls the flexibility vs brittleness of the stream model. At one extreme, small λ values result in frequently rebuilding the model and including noise patterns in it; at the other extreme, large λ values lead to discard the first representative of new clusters and cause the stream model to lag behind the data distribution. Therefore, an appropriate setting of λ is desired to enable the stream model to keep pace with changes in data distribution. A problem is that λ should depend on the scale of the observed phenomenon, but the triggering of the rebuilding should not. In this paper, we propose a self-calibrating definition of the threshold, instead of a fixed pre-defined value as in [15] or the optimized value under the Bayesian Information Criterion related to the stream model [16]. From Equation 7, we see that $PH_t \approx (t-t_0)(\bar{p}_{t_0} - \mu_1 - \delta)$. As we are tracking situations where μ_1 becomes small, and δ is always negligible, the threshold can be set as:

$$\lambda_{t_0} = \begin{cases} 0 & \text{if } PH_t = 0 \\ f * \bar{p}_{t_0} & \text{otherwise,} \end{cases} \quad (8)$$

where f is a constant called the λ-factor. f is the "critical mass" of a new cluster, and can be interpreted as the number of required witnesses seeing the changes. This way, the change point detection is scale-invariant. In order to cope with the variability of the p_t, the threshold can be evaluated at each time step, as $f * \bar{p}_t$. The sensitivity analysis w.r.t. the λ-factor, f, is presented in Section 5.3.

4.3 Model Rebuilding

Upon triggering the change point detection test (or when the reservoir is full), a new model is rebuilt by launching Weighted AP on the dataset $\mathcal{E}$ made of the model exemplars together with their number of represented items (e_i, n_i) and the outliers in the reservoir $(x_j, 1)$. Along the same lines as discussed in Section 3.2, the energy terms are defined as:

$$\begin{array}{lll} S(e_i,e_i) & = \sigma + \Sigma_i & S(x_j,x_j) \quad = \sigma \\ S(e_i,e_j) & = -n_i\, d(e_i,e_j)^2 & \\ S(e_i,x_j) & = -n_i\, d(e_i,x_j)^2 & \\ S(x_j,e_i) & = -d(x_j,e_i)^2 & \end{array} \quad (9)$$

Note that S is asymmetric. The cost of selecting current exemplar e_i to be the exemplar of x_j is ordinary similarity $-d(x_j,e_i)^2$, while the cost of selecting x_j to be the exemplar of e_i is increased by a factor of n_i. Therefore, the current exemplar e_i will have more chance to be an exemplar again. As discussed in Section 3.2, the optimization problem defined by this asymmetric matrix is equivalent to the optimization problem defined on the whole data by unfolding each exemplar e_i to all of its associated items.

WAP accordingly selects the optimal exemplars in $\mathcal{E}$. The new stream model is finally defined as follows. Let $\mathring{e}$ denote a new exemplar representing previous exemplars $e_1, \ldots, e_m$ and outliers $x_1, \ldots, x_{m'}$. With no difficulty, the number n of items associated to $\mathring{e}$ is set to $n_1 + \ldots + n_m + m'$. The associated distortion Σ is estimated as follows. Let x be an item associated to e_1. Indeed x is no longer available after visiting the streaming model; but assuming an Euclidean space, x can be modeled as a random item $e_1 + Xv$, where v is a random vector in the unit sphere, and X is a scalar random variable. It comes:

$$d(\mathring{e}-x)^2 \quad = d(\mathring{e}-e_1)^2 + d(e_1-x)^2 - 2X\langle \mathring{e}-e_1, v\rangle$$

Assuming that v is uniformly distributed in the unit sphere, and that the distribution of the norm and the angle are independent, summing over x in the cluster represented by exemplar e_1 and taking the expectation give: $\mathbb{E}[\sum_{x\in C_{e_1}} d(\mathring{e},x)^2] = n_1 d(\mathring{e},e_1)^2 + \Sigma_1$, as the expectation of the scalar product w.r.t the uniform distribution is 0. Accordingly,

$$\Sigma = \sum_{i=1}^{m}\left(n_i d(\mathring{e},e_i)^2 + \Sigma_i\right) + \sum_{i=1}^{m'} d(\mathring{e},x_i)^2$$

Finally, t is set to the maximal time stamp associated to e_i and x_j, for e_i and x_j ranging among the exemplars and outliers associated to $\mathring{e}$.

4.4 Theoretical analysis of distortion loss in minor exemplar drift

Exemplars in StrAP are used to summarize streaming items and reconstruct models, as introduced in Section 4.1 and 4.3. The new exemplars after model

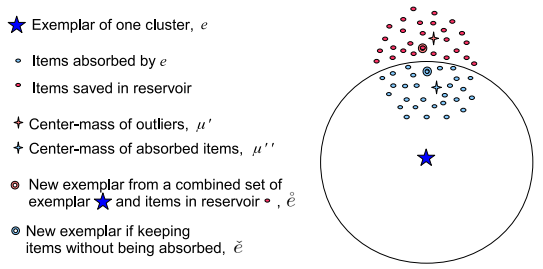


Fig. 2. Illustration of exemplars built with and without discarded items inside the ε–neighborhood of an exemplar e

rebuilding are either old exemplars or items in reservoir, as items inside ε–neighborhood of exemplars are discarded when they flowed in and were absorbed by their respective nearest exemplars. However, the optimal exemplars may drift in streaming. A significant drift can be caught by our change detection and rebuilding method. A minor drift within ε can be coped with by retaining the current exemplar and adapting the corresponding model parameters, or by selecting in reservoir a new exemplar close to the current exemplar. For example in Figure 2, an exemplar e drifts to $\check{e}$, which has a distance less than ε to e. Items of new distribution with $\check{e}$ are then partially absorbed by e (including $\check{e}$ itself), and partially reported as outliers in reservoir. After model rebuilding, the new exemplar will be either the previous e or a new one $\mathring{e}$ from reservoir since $\check{e}$ is absent. The optimal exemplar $\check{e}$ achieves the minimum distortion, while e and $\mathring{e}$ lead to an increase of distortion. In this section, we theoretically analyze the distortion loss caused by discarding items after updating the model when minor exemplar drift happens.

Assume that all items from the distribution with exemplar $\check{e}$ are composed of m' outside ($\mathcal{E}_{m'}$, in reservoir) and m'' inside ε–neighborhood of e ($\mathcal{E}_{m''}$). If keeping all of them available, $\check{e}$ is extracted from a combination of e and $m'+m''$ items with the optimal distortion:

$$D_{\check{e}} = \sum_{x_i \in \mathcal{E}_{m'}} \|x_i - \check{e}\|^2 + \sum_{x_i \in \mathcal{E}_{m''}} \|x_i - \check{e}\|^2 + n\|e - \check{e}\|^2 + \Sigma_e$$

where n and Σ_e are model parameters w.r.t. e without absorbing m'' items.

According to the update equation of distortion Σ in Section 4.3, if m'' items were absorbed by e, a new $\Sigma_{e+\mathcal{E}_{m''}}$ is obtained: $(\frac{\Delta}{\Delta+1})^{m''}\Sigma_e + \sum_{i=1, x_i \in \mathcal{E}_{m''}}^{m''} (\frac{\Delta}{\Delta+1})^{m''-i} \frac{n_i}{n_i+1}\|x_i - e\|^2$. In the case that e is selected again as the new exemplar from a combination of e and $\mathcal{E}_{m'}$, the distortion is:

$$D_e = \sum_{x_i \in \mathcal{E}_{m'}} \|x_i - e\|^2 + \Sigma_{e+\mathcal{E}_{m''}}$$

In the case that $\mathring{e}$ is selected as the new exemplar from a combination of e and $\mathcal{E}_{m'}$, the distortion is:

$$D_{\mathring{e}} = \sum_{x_i \in \mathcal{E}_{m'}} \|x_i - \mathring{e}\|^2 + \Sigma_{e+\mathcal{E}_{m''}} + (n+m'')\|e - \mathring{e}\|^2$$

We first verify that the distortion loss in the first case selecting e as the new exemplar is negligible.

$$\begin{aligned} D_e - D_{\check{e}} &= \sum_{x_i \in \mathcal{E}_{m'}} (\|x_i - e\|^2 - \|x_i - \check{e}\|^2) - n\|e - \check{e}\|^2 \\ &\quad + [\Sigma_{e+\mathcal{E}_{m''}} - \Sigma_e - \sum_{x_i \in \mathcal{E}_{m''}} \|x_i - \check{e}\|^2] \end{aligned} \tag{10}$$

Let $\mu' = \frac{1}{m'}\sum_{x_i \in \mathcal{E}_{m'}} x_i$ be the center-mass of $\mathcal{E}_{m'}$, and $\mu'' = \frac{1}{m''}\sum_{x_i \in \mathcal{E}_{m''}} x_i$ be the center-mass of $\mathcal{E}_{m''}$. We define $a = e - \mu'$, $b' = \check{e} - \mu'$, $a' = \mathring{e} - \mu'$, $a'' = e - \mu''$ and $b'' = \check{e} - \mu''$. We have

$$\begin{aligned} &\textstyle\sum_{x_i \in \mathcal{E}_{m'}} (\|x_i - e\|^2 - \|x_i - \check{e}\|^2) \\ &= m'(e - \check{e}) \cdot (e + \check{e} - 2\mu') = m'(\|a\|^2 - \|b'\|^2) \end{aligned} \tag{11}$$

and

$$\begin{aligned} &\Sigma_{e+\mathcal{E}_{m''}} - \Sigma_e - \sum_{x_i \in \mathcal{E}_{m''}} \|x_i - \check{e}\|^2 = \left((\tfrac{\Delta}{\Delta+1})^{m''} - 1\right)\Sigma_e \\ &+ \sum_{\substack{i=1, \\ x_i \in \mathcal{E}_{m''}}}^{m''} \left((\frac{\Delta}{\Delta+1})^{m''-i} \frac{n_i}{n_i+1}\|x_i - e\|^2 - \|x_i - \check{e}\|^2\right) \\ &< \sum_{x_i \in \mathcal{E}_{m''}} \|x_i - e\|^2 - \|x_i - \check{e}\|^2 = m''(\|a''\|^2 - \|b''\|^2) \end{aligned} \tag{12}$$

Taking Eq. (11), and (12) into Eq. (10), we have

$$\begin{aligned} &D_e - D_{\check{e}} < m'(\|a\|^2 - \|b'\|^2) \\ &+ m''(\|a''\|^2 - \|b''\|^2) - n(\|a\|^2 + \|b'\|^2) + 2n\|a\|\|b'\| \end{aligned}$$

Considering that $\|a''\| < \|a\|$ (e is closer to μ'' than to μ'), $\|b''\| < \|b'\|$ ($\check{e}$ is closer to μ'' than to μ'), and $\|b'\| < \|a\|$ (μ' is closer to $\check{e}$ than to e), we can have

$$D_e - D_{\check{e}} < (m' + m'' + n)(\|a\|^2 - \|b''\|^2)$$

Since $\check{e}$ is the optimal exemplar, center-mass μ' and μ'' are near to $\check{e}$. Hence, $\|b''\|$ is close to 0, and $\|a\|$ is close to ε. Then, we can have the distortion loss of $D_e - D_{\check{e}}$ bounded by $(m' + m'' + n)\varepsilon^2$. The distortion loss per item is ε^2, which is usually small and is acceptable.

We now verify the distortion loss in the second case selecting $\mathring{e}$ as the new exemplar.

$$\begin{aligned} D_{\mathring{e}} - D_{\check{e}} &= \sum_{x_i \in \mathcal{E}_{m'}} (\|x_i - \mathring{e}\|^2 - \|x_i - \check{e}\|^2) - n\|e - \check{e}\|^2 \\ &+ [\Sigma_{e+\mathcal{E}_{m''}} - \Sigma_e - \sum_{x_i \in \mathcal{E}_{m''}} \|x_i - \check{e}\|^2] + (n+m'')\|e - \mathring{e}\|^2 \end{aligned} \tag{13}$$

Similar to the first case, we can have

$$\begin{aligned} D_{\mathring{e}} - D_{\check{e}} &< (m'+n)(\|a'\|^2 - \|b'\|^2) + 2na \cdot (b' - a') \\ &+ m''(\|a''\|^2 - \|b''\|^2) + m''(a - a')^2 \end{aligned}$$

Since $\|a'\| < \|b'\|$ (μ' is closer to $\mathring{e}$ than to $\check{e}$), $a - a' = e - \mathring{e}$, $\|a''\|^2 < \varepsilon^2$ and $\|b' - a'\| = \|\check{e} - \mathring{e}\|$, it comes

$$D_{\mathring{e}} - D_{\check{e}} < m''(\|e - \mathring{e}\|^2 + \varepsilon^2) + 2n\|a\|\|\check{e} - \mathring{e}\|$$

Since $\mathring{e}$ is near to the optimal exemplar $\check{e}$ ($\|\check{e} - \mathring{e}\|$ is close to 0), $\|e - \mathring{e}\|$ and $\|a\|$ are close to ε. Then, the upper bound of $D_{\mathring{e}} - D_{\check{e}}$ is around $2m''\varepsilon^2$, which is an acceptable distortion loss.

4.5 Memory Usage and Complexity Analysis

Streaming algorithms are required to have a fixed and small memory usage and a short computing time in the whole process. The memory usage of STRAP algorithm mainly consists of all exemplars in model $(e_i, n_i, \Sigma_i, t_i)$ and the outliers in the reservoir. The statistical variables in change detection consumes ignorable memory. Since the number of exemplars is usually small and the reservoir has a limited size, the overall memory usage is much smaller than the load of intuitively keeping in memory all items in a sliding window. The memory usage varies a little in the streaming process. More exemplars are included in the model when streaming data are from a more complex underlying distribution. However, memory is released by eliminating out-of-date exemplars so that the model would not get ponderous. Keeping only recent and active exemplars is also beneficial to the computational complexity, which is analyzed in the next paragraph.

Suppose there are m exemplars in the current model and m' outliers in the reservoir. At each time t when a new item arrives, the updating operation in Section 4.1 has linear complexity w.r.t. m. The change detection method in Section 4.2 calculates p_t, $\bar{p}_t$, m_t, M_t and λ_t by simple operations like summation and multiplication on results from last time step, and thus takes a constant computing time. The model rebuilding in Section 4.3 applies WAP and updates model $(e_i, n_i, \Sigma_i, t_i)$ on a combination of m exemplars and m' reservoir items. The complexity of this step is $\mathcal{O}((m + m')^2)$. The overall time complexity of STRAP algorithm is quadratic w.r.t. the number of exemplars and reservoir items. Eliminating outdated elements from the model and reservoir as aforementioned improves the time complexity of STRAP algorithm.

The experimental results of memory usage and computing time are reported in Table 2 and 3.

5 Experimental Validation

This section reports the empirical validation of STRAP algorithm on two benchmark data sets and the application to a real grid monitoring system. Let us first present the assessment criteria before detailing the experimental settings and discussing the results.

5.1 Assessment criteria

The streaming model is evaluated on two aspects, *quality* and *time-effectiveness*. With the help of external class labels, the quality of online clustering is measured by clustering accuracy and purity. When assessing clustering accuracy, the labels of items grouped in one cluster are determined by the class label of their exemplar. An item is misclassified iff it is associated to an exemplar belonging to a different class. In the following, the *clustering accuracy* stands for the percentage of data items that are correctly classified.

The clustering accuracy criterion however poorly accounts for the case of imbalanced datasets and the correct classification of examples in the minority class(es). A second criterion, referred to as *clustering purity*, is thus used. It averages the fraction of items belonging to the majority class of in each cluster.

The third criterion, the *percentage of outliers*, assesses the time-effectiveness of the model. By construction, outliers are far from the current model and their information is thus poorly accounted for; this makes it desirable to keep the number of outliers as low as possible. On the other hand, a healthy fraction of outliers contributes to keeping the computational load and the number of model rebuilt within reasonable limits; furthermore, the outliers in the reservoir are used to seed the new model with relevant information about the new clusters.

5.2 Experimental setting

The goal of the experiments is to comparatively assess STRAP w.r.t. all above three criteria. Two data sets, the thoroughly studied Intrusion Detection benchmark (KDD'99) [31], [32] and a recent streaming data of benign and malicious URLs [19], are employed for the evaluation.

The applicative goal of KDD'99 is to tell attacks from normal connections. A connection is a sequence of TCP packets starting and ending at some well specified times, flowing from a source IP address to a target IP address under well defined protocols. It is described by 41 attributes; following [9] we only used the 34 numeric ones and normalized them[4].

Each connection is labeled among 23 classes, the *normal* class and the specific kinds of attack, such as *buffer_overflow*, *ftp_write*, *guess_passwd*, and *neptune*. The relevance of this data from a stream perspective comes from the fact that it involves a non-stationary data distribution: attacks evolve along a typical arm race between the system administrators and the intruders; and the normal usages also evolve. STRAP is expected to build a dynamic model of the different types of connections, as they appear in the flow. Each cluster would include connections of a same type.

The streaming URLs are studied to predict malicious URLs from benign URLs [19], and thus protect users from accessing on rogue Web sites. The 120-day streaming data consists of more than 2 million URLs, each of which is described by lexical features, e.g., the unusual appearance of '.com' in the middle of

4. Attribute *duration* is changed from seconds into minutes; *src_bytes* and *dst_bytes* are converted from byte to KB; *log* is used on *count*, *srv_count*, *dst_host_count*, *dst_host_srv_count*.

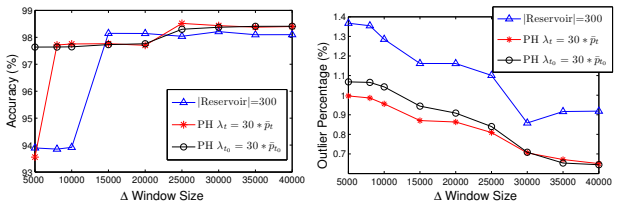
a URL, and host-based features, e.g., the ownership and IP prefix of a web site host. A URL is labeled as benign or malicious. Ma et al. in [19] designed an online supervised learning algorithm, which has a high accuracy of 98-99%.

We applied our algorithm on clustering the URLs stream to produce pure clusters of benign or malicious URLs sharing similar feature values. Newly coming URLs would thus be classified after the type of the exemplar they are associated to, or identified as outliers (and thus be considered as malicious ones). To make an efficient clustering, we used the 64 numeric attributes among millions of attributes.

The initialization of the stream model (Algorithm 1) considers the first 1000 connections of the KDD'99 data set which totally includes 494,021 network connection records (71MB), and the first 1000 URLs of the 2,396,130 URLs data set (500MB).

5.3 Sensitivity analysis

Beyond measuring the three performance criteria, we also intend to measure the sensitivity of the results w.r.t. the STRAP parameters: the window length Δ (Section 4.1) and the restart triggering parameters (Figure 3, 4, 5 and Table 2).

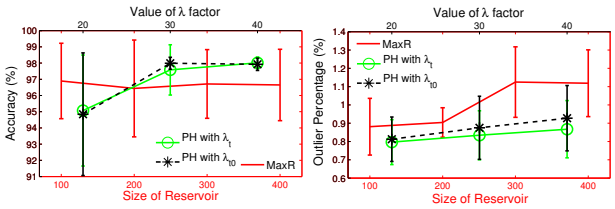


(a) Clustering accuracy (b) Percentage of outliers

Fig. 3. Performance of STRAP on KDD'99 dataset: comparing restart criterion PH ($f = 30$) and MaxR ($|Reservoir| = 300$), depending on window length Δ

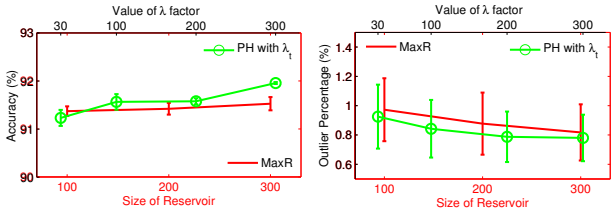
Figure 3 compares the performance of STRAP on KDD'99 data depending on the window length Δ. Figure 3 (a) shows that STRAP achieves high accuracy (more than 97% for $\Delta > 15000$). The percentage of outliers of PH is 0.3% (around 1500 network connections) lower than that of MaxR, as shown in Figure 3 (b). The two ways of setting λ for PH criterion have comparable impact on clustering accuracy. PH with λ_{t_0} has a marginally higher percentage of outliers.

Figure 4 and 5 displays the influence of the restart parameters in KDD'99 and ULRs data. On each of different settings on $|Reservoir|$ (bottom x axis in Figure 4 and 5), STRAP was run several times with different window length of Δ. The average performance and std are computed on the results of independent runnings. Likewise, the sensitivity w.r.t. the PH parameters (settings on λ_t factor f at the top x axis) is illustrated from the average and std of results with different values of Δ.

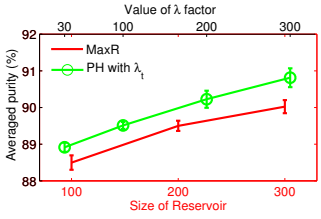


(a) Clustering accuracy (b) Percentage of outlier

Fig. 4. Performance of STRAP on KDD'99 dataset: comparing restart criterion PH and Reservoir size depending on parameter setting



(a) Clustering accuracy (b) Percentage of outlier



(c) Averaged clustering purity

Fig. 5. Performance of STRAP on URLs streams: comparing restart criterion PH and Reservoir size depending on parameter setting

Figure 4 (a) and 5 (a) shows the average accuracy and Figure 4 (b) and 5 (b) shows the average percentage of outliers. Figure 5 (c) shows the average clustering purity. We can see MaxR has stable accuracy on the $|Reservoir|$ setting and causes more outliers with the increasing of $|Reservoir|$. PH criterion has higher accuracy when $f \geq 30$, lower percentage of outliers and also higher purity.

Table 2 gives the computational cost[5] and memory usage of STRAP on various parameter settings, focusing on the most representative results. Note that the presented results are measured on clustering the whole data stream. Under the value of *time cost/memory usage*, we also give the range of number of clusters during the stream clustering process. The memory usage, which is mainly proportional to the number of clusters, varies along time. We thus give its mean with std in the whole process.

The computational cost increases with Δ; when using MaxR as restart criterion, the computational

5. measured on an Intel 2.66GHz Dual-Core PC with 4 GB memory for KDD'99 (for being comparable to results in [9] obtained on a 3.4 GHz PentiumIV PC with 1GB memory) and a Dell T7500 workstation which has 6 Intel Xeon processor X5650 2.67GHz with 12 cores and 24GB of RAM for URLs data.

cost also increases with the size of the reservoir. Both effects are blamed on the fact that larger reservoir size and Δ values entail that the model is rebuilt from a larger dataset and yield more clusters (Section 4.3). In contrast, the computational cost decreases by 10% on average (more significant on URLs) when using the PH restart criterion; this reduction is explained as the PH criterion fires when new patterns arrived and enough examples of them are collected, thus avoiding to take irrelevant clusters in the model. The small amount of memory usage is almost fixed (with small variances), which verifies the analysis in section 4.5.

TABLE 2
Time cost/memory usage (N. of evolving clusters) of STRAP on different parameter settings when applying to the whole KDD'99 and URLs stream (in *mins/MB*)

(a) KDD'99

Restart	Δ=5000	Δ=10000	Δ=20000	Δ=30000
PH $\lambda = 30 * \bar{p}_t$	**3.2/0.7±0.3** ([13 154])	**3.7/0.7±0.4** ([36 184])	**4.1/0.9±0.5** ([19 197])	**4.8/1.1±0.5** ([57 215])
MaxR =100	**3.2/0.7±0.3** ([18 154])	**3.8/0.7±0.4** ([16 188])	**4.2/0.8±0.5** ([17 228])	**4.5/1.0±0.5** ([57 248])
MaxR =300	**3.5/0.7±0.3** ([31 181])	**4.4/0.7±0.3** ([20 224])	**5.2/0.7±0.3** ([48 239])	**5.2/0.7±0.3** ([57 244])

(b) URLs

Restart	Δ=1000	Δ=5000	Δ=10000	Δ=20000
PH $\lambda = 30 * \bar{p}_t$	**21/3.3±0.1** ([47 195])	**23/3.4±0.1** ([58 191])	**28/3.5±0.2** ([58 249])	**36/3.7±0.2** ([58 297])
MaxR =300	**23/3.4±0.1** ([58 222])	**26/3.5±0.2** ([58 203])	**30/3.6±0.2** ([58 260])	**47/3.8±0.3** ([58 322])

Summarizing Figure 4, 5 and Table 2, the PH criterion improves on the MaxR criterion, with a higher accuracy and purity (1%), a lower percentage of outliers (0.2%) and a smaller computational time (10%). It is worth noting that STRAP only needs 1% of the data (initial subset plus the outliers) in order to produce an accurate model (more than 97% accuracy) in KDD'99. Considering the KDD'99 data as a binary classification problem, this problem includes about 20% normal connections and 80% attacks. The online clustering results of STRAP yield 99.18% True Detection rate and 1.39% False Alarm rate, to be compared with [33] using a supervised method and yielding 98.8% True Detection rate and 0.4% False Alarm rate. STRAP achieves an accuracy around 91-92% for URLs data, which is not comparable to that obtained by supervised algorithms in [19] considering that STRAP is unsupervised. Please also note we only used 64 numeric attributes rather than all the millions of attributes as used in [19].

5.4 Online performance comparison

The online performance of STRAP on KDD'99 is compared to *DenStream* in Figure 6. The clustering purity of *DenStream* on the KDD'99 dataset was evaluated during four time windows of length 1000 when some attacks happened. For a fair comparison, the clustering purity of STRAP was computed during the same time windows, considering the same 23 classes. Figure 6 respectively displays the STRAP results obtained for one of the worst settings ($\Delta = 10000$ and $|Reservoir| = 300$) and an average setting ($\Delta = 10000$ and $\lambda_{t_0} = 30 * \bar{p}_{t_0}$), together with the *DenStream* results reported from [9]. STRAP outperforms *DenStream* on all assessed windows.

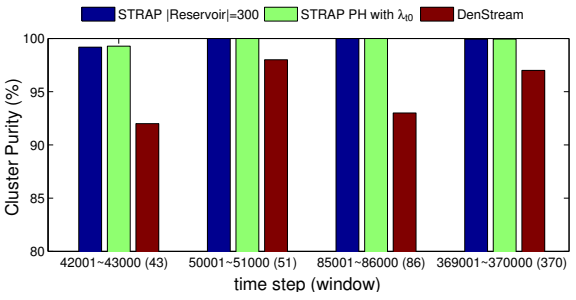


Fig. 6. Comparative performances of STRAP and *DenStream* on Intrusion Detection dataset

Figure 7 displays the accuracy along time of STRAP on URLs data for $\Delta = 10000$ and $\lambda_t = 300 * \bar{p}_t$ in PH criterion. Each accuracy rate measured in window of 10000 evaluates the clustering quality for URLs in around half a day. It can be observed that the online clustering accuracy is above 91% in most of the time. Figure 8 shows the online clustering purity of STRAP. After each model rebuilt, the purity is computed by averaging the purity in each cluster. The number of clusters K is also reported (as increasing K would mechanically improve the purity), showing that K remains circa 300 and yields a purity above 90% in most of the time.

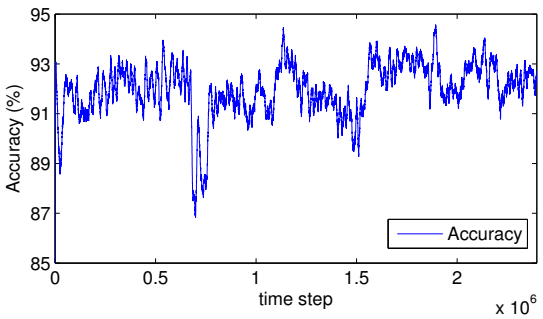


Fig. 7. Online clustering accuracy of STRAP on URLs data set when $\Delta = 10000$ and λ_t factor $f = 300$

5.5 Application to Grid Monitoring

As presented in the introduction, the motivating application of STRAP is to provide the Grid system administrator with a dashboard monitoring the current status of the system, for the early detection of undesired events.

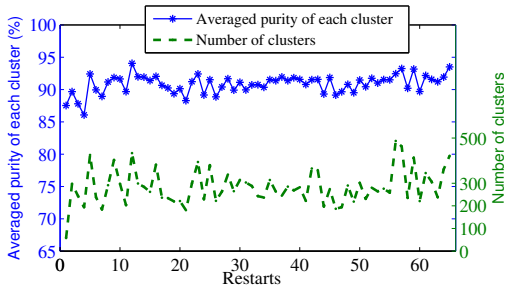


Fig. 8. Online clustering purity of STRAP on URLs data set when $\Delta = 10000$ and λ_t factor $f = 300$

In the EGEE/EGI grid, Resource Brokers (RB) are the software components that dispatch the incoming jobs to the queues of the local batch systems. 5,268,564 EGEE jobs are extracted from the logs of 39 Research Brokers of all gLite-operated jobs in the whole EGEE grid during 5 months, from 2006-01-01 to 2006-05-31.

In the following, each job is described by 6 attributes measuring the time duration for different services. During the job lifecycle, a job is first *submitted*, then *waiting* for the Workload Management System (WMS) to match a resource for it. Once a resource is found, the job becomes *ready* for transfer; it is thereafter transferred to the resource, where its state is *scheduled*, meaning that it is enqueued in the local batch system. When selected, the job is *running*, until successfully finished (*done OK*), or failed (*done failed*). Accordingly, the six time duration attributes are:

1) $T_{submission}$: time between job registration and transfer to WMS
2) $T_{waiting}$: time to find a matching resource
3) $T_{transfer}$: time acceptation and transfer (*waiting* + *ready* time), reported by the JobController (JC)
4) T_{CE_accept}: the same as $T_{transfer}$, but reported by the LogMonitor (LM)
5) $T_{scheduled}$: queuing delay
6) $T_{running}$: execution time.

Note that attributes 3 and 4 are functionally redundant: JC is a standalone logging service, while the LM integrates various logs, and returns them in the logging and bookkeeping (L&B) database; we shall nevertheless see that their discrepancies offer valuable insights into the state of the system.

All attributes differ by orders of magnitude; they are thus normalized $x \to \frac{x-\mu}{s}$ where μ (respectively s) denotes the mean (respectively std) of attribute x measured on the initial bunch of data (set to the first 1000 jobs;). Furthermore, in case the job fails at some point along its lifecycle, the durations associated to the subsequent services are not available and set to 0 by default. Six additional boolean attributes are thus also considered, indicating whether the job reaches any of the six states. Finally, the similarity between two jobs is set to the Euclidean distance on $\mathbb{R}^{12}$.

Besides the descriptive attributes, each job is labeled after its final state, successfully finished (*good job*) or failed (*bad job*). The failure cases are further categorized into 45 error types, e.g., *Cancel requested by WorkloadManager*, *RB Cannot plan*. The 5 million samples considered in the experiments involve 20 main error types, including more than 1,500 occurrences each.

STRAP is applied to the job data flow, after their description has been normalized. Its control parameters are set as follows. STRAP model is initialized by launching AP on the first 1,000 jobs. The preference penalty σ in AP is set to the median value of similarity matrix. The window length Δ used in STRAP is set to $10,000$. The parameters of the PH restart criterion are $\delta = 0.01$ and $\lambda_t = 30 * \bar{p}_t$.

5.5.1 Online monitoring snapshots

The output of STRAP is displayed on Figure 9, showing two snapshots at different time. Each snapshot illustrates the data distribution as a histogram; each bar stands for a cluster, topped with the exemplar description and whose height is the fraction of jobs attached to this exemplar since the model rebuilt. For the sake of readability, only clusters with sufficient representativity (more than 1%) are represented.

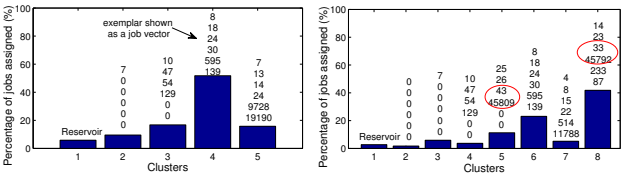


Fig. 9. Snapshots from the monitoring output

The left snapshot of Figure 9 illustrates a standard regime; 60% of jobs are successfully finished, with typical time profile "[8 18 24 30 595 139]". A small fraction of successful jobs are computationally heavy jobs (execution time circa 19190s) and they spent a long time waiting in the queue (circa 9728s), 10% of the jobs (exemplar [7 0 0 0 0 0]) fail almost immediately (they stop after their registration); 20% of the jobs (exemplar [10 47 54 129 0 0]) fail at a later stage, before arriving at the local computing site.

The right snapshot of Figure 9, recorded 3 days later, illustrates an emergency situation. It includes clusters similar to those in the left snapshot, plus two untypical clusters. Cluster 8, which includes about 40% of the jobs (exemplar [14 23 33 45792 233 87]), is considered to be anomalous as its T_{CE_accept} is very long. This situation is interpreted as the LogMonitor (LM) becoming clogged, with a typical LM-time value of 45792s, compared to tens of seconds in the former snapshots. This conjecture is supported by the LM-time in Cluster 5 (exemplar "[25 26 43 45809 0 0]"), where jobs fail before arriving at the local site (likewise the cluster with exemplar [10 47 54 129 0 0]), but has much larger LM-time. Interestingly, this problem was discovered without using the job labels; and no failure case corresponds to the clogging of the LM.

The online monitoring thus yields a compact and intuitive summary of the job distribution, which can be interpreted on the spot by the system administrator. As will be seen (Figure 9), the system also provides a day-, week- and month-view of the job data flow[6]. The processing meets the real-time requirements (40,000 jobs are processed in 1 min).

5.5.2 Online clustering accuracy, purity and efficiency

In order for the stream model to be usable, it is desirable that the clusters are pure w.r.t. the job labels to the best possible extent. Figure 10 reports the online accuracy of the clusters w.r.t. the job labels, together with the percentage of outliers, depending on the restart parameters.

The baseline results are provided by a *Streaming k-medoids* algorithm, replacing the AP component in STRAP with a k-medoids algorithm, and retaining the best results out of 20 independent runs; in this way, AP and k-medoids require the same computational time. When rebuilding the model, k-medoids is launched on a dataset gathering copies of the model and items of the reservoir. To imitate WAP in STRAP, k-medoids is used with memory in which 80% of the initially selected of items are from the model and others are from the reservoir. The number k of clusters in k-medoids is set to 133, which is the average number of exemplars in STRAP.

Note that the use of *DenStream* was excluded as the *DenStream* model is only made explicit upon request. Requiring it to classify every job and demonstrating each cluster by a representative exemplar were beyond the available computational resources.

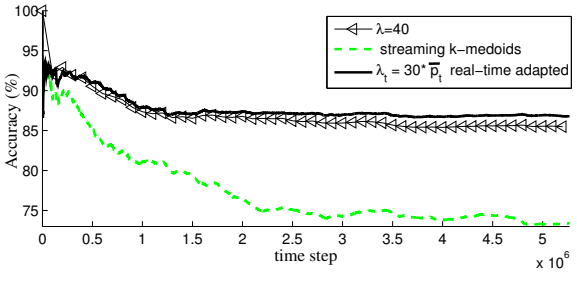


Fig. 10. Online clustering accuracy when summarizing the job flow of EGEE, in comparison among STRAP with real-time adapted $\lambda_t = 30 * \bar{p}_t$, STRAP with λ fixed by a given value 40, and Streaming k-medoids

Figure 10 shows that STRAP improves on Streaming k-medoids by circa 5%, while involving circa 4% less outliers. The restart parameters with real-time adapted λ_t were found to provide more stable and satisfactory results than the fixed λ used in our previous work [16].

Likewise, Figure 11 shows the average clustering purity over all clusters after each model rebuilt; the number k of clusters is also depicted, to confirm that the high purity cannot be attributed by an extra-large number of clusters. Interestingly, the purity is circa 90% for $k = 200$ clusters on average, and is thus higher than the clustering accuracy; this higher purity is attributed to the fact that some large clusters are mixed, while many small clusters are pure.

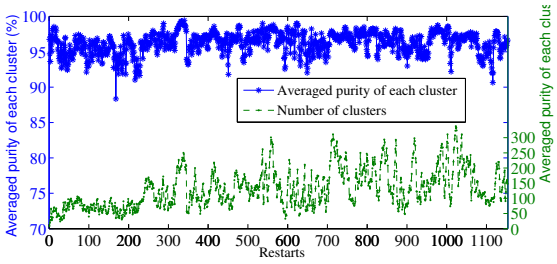


Fig. 11. Clustering purity after each restart

The running time and memory usage on EGEE job stream are shown in Table 3 (run on the workstation). The high computing cost (more than an hour for clustering more than 5 millions jobs) is mainly due to the long string labels in the data set. It took more time to update clusters with long strings than numeric/binary labels.

TABLE 3
Time cost/memory usage (N. of evolving clusters) of STRAP on different parameter settings when applying to the whole EGEE job stream (in *mins/MB*)

Restart	Δ=5000	Δ=10000	Δ=20000
PH $\lambda = 30 * \bar{p}_t$	**71/1.8±0.4** ([9 467])	**73/1.9±0.4** ([9 431])	**96/2.2±0.5** ([9 544])
MaxR =300	**55/1.6±0.2** ([9 275])	**65/1.8±0.3** ([9 390])	**110/2.1±0.5** ([9 499])

5.6 Discussion

While many data streaming algorithms actually focus on the extraction of statistical information from data streams [34], [35], [36], ranging from the approximation of frequent patterns [37] to the construction of decision trees [38], the most related work is [9], similarly addressing unsupervised learning and clustering from data streams. The *DenStream* algorithm upgrades the *DbScan* clustering algorithm [27] to dynamic environments; it mainly differs from STRAP regarding the creation and the updating of clusters. Actually, *DenStream* does not construct the final clusters unless requested to do so by the user; upon such a request, the (most recent) items will be labeled after the clusters. While this "lazy" clustering and labeling behavior is more computationally efficient, it is suggested that it is not well-suited to e.g., monitoring applications, when the goal is to identify behavioral drifts as soon as they appear (Section 5.5.1).

Comparatively to *DenStream*, STRAP provides a model at any time step; further, this model was

6. Other monitoring results in avi format are provided at http://www.lri.fr/~xlzhang/GridMonitor/.

shown to be more accurate than *DenStream* on KDD'99 dataset. In counterpart, STRAP computational time is higher than *DenStream* (4 minutes against 7 seconds). A question opened for further investigation is whether this performance is compatible with real-time applications; an alternative is to design a distributed version of STRAP, e.g., sharing the reservoir.

Another relevant work, presented by Cormode et al. [39], aims at the structure of clusters in the stream. Interestingly, an extension of AP referred to as Soft-Constraint AP (SCAP) has been proposed by Leone et al. [40]. Further research will investigate the extension of SCAP to data streaming, to address the structured cluster extraction from data streams.

The exemplars built on the fly by STRAP can be used to provide a retrospective and macro-scale view of the streaming data. "Super-exemplars" can be extracted by applying AP on all exemplars selected during the target period (e.g., 5 months in the past). These super-exemplars provide a common reference, enabling to describe the overall distribution observed in that period, and thus to detect the long-run trends.

6 CONCLUSION

Our proposed algorithm STRAP aims at clustering data streams with evolving data distributions. STRAP confronts the arriving items to the current AP model, storing the outliers in a reservoir and monitoring the ratio of outliers using the PH change point detection test. Upon triggering the PH test, the clustering model is rebuilt from the current one and the reservoir using WAP (Section 4.3). The key issue here was to build the change indicator, monitored by the PH test, in order to preserve the computational cost vs accuracy tradeoff. In this paper, we monitored the ratio of outliers over a sliding window and adapted the change detection threshold in real-time (Section 4.2).

The proposed approach STRAP was theoretically analyzed on guaranteeing acceptable distortion loss when exemplars slightly drift from the already selected ones, on consuming small amount of memory with little variation, and on requiring acceptable computing time that depends on the complexity of underlying distribution. The performance of STRAP in clustering quality and efficiency is empirically validated on KDD'99 benchmark problem and the URLs stream. While STRAP improves on *DenStream* w.r.t. clustering purity (shown in Figure 6), it is slower; the interpretation offered for this fact is that STRAP provides a model of the stream at any time step, whereas *DenStream* only builds the model upon request. The validation on URLs stream also demonstrates that STRAP achieves above 90% accuracy and purity for grouping benign and malicious URLs, given that STRAP is unsupervised and worked on only 64 out of millions of attributes.

The last contribution of this paper is the application of STRAP on monitoring of the grid job streams motivated by the field of Autonomic Grid. STRAP is used on a 5-million job traces, and it showed the feasibility of providing the grid administrators with a real-time dashboard of the job data flow. This online report enables the instant detection of regime drifts (e.g., clogging of LogMonitor as shown in Figure 9).

This work opens several perspectives for further research. A recurrent question for clustering problems is to define the "natural" number of clusters, or for AP the best value of the penalty parameter σ. Our recent work has shown how to directly generate a given number of clusters by modifying the AP formulation [41]. Another perspective is to organize the streaming model in a hierarchical manner, to speed up the retrieval phase in STRAP. Presently, each new item is confronted to all exemplars; it would be thus desirable to organize these exemplars, and investigate how e.g., Locality Sensitive Hashing [42] can support the pre-selection of the nearest exemplars. Last, in the Autonomic Grid application, clustering can be used to group users, who are represented by the exemplars of their submitted jobs. These clusters will be instrumental in defining user-friendly grid interfaces.

ACKNOWLEDGMENTS

We thank Dr. Julien Perez for valuable discussions about the parametrization of the change point detection test. This work has been partially supported by the EGEE-III project funded by the European Union INFSO-RI-222667 and supported by Network of Excellence PASCAL, IST-2002-506778.

REFERENCES

[1] J. Gama and M. M. Gaber, *Learning from Data Streams: Processing Techniques in Sensor Networks*. Springer, December 2007.
[2] C. Aggarwal, *Data Streams: Models and Algorithms*. Springer, 2007.
[3] G. Cormode, S. Muthukrishnan, and W. Zhuang, "Conquering the divide: Continuous clustering of distributed data streams," in *ICDE*, 2007, pp. 1036–1045.
[4] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams: Theory and practice," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 15, pp. 515–528, 2003.
[5] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," in *IEEE Symposium on Foundations of Computer Science*, 2000, pp. 359–366.
[6] C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *VLDB*, 2003, pp. 81–92.
[7] M. R. Ackermann, M. Maartens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "StreamKM++: A clustering algorithm for data streams," *CM Journal on Experimental Algorithmics*, vol. 17, no. 1, pp. 2.4:2.1–2.4:2.30, May 2012.
[8] M. Shindler, A. Wong, and A. Meyerson, "Fast and accurate k-means for large datasets," in *NIPS*, 2011, pp. 2375–2383.
[9] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *SDM*, 2006, pp. 326–337.
[10] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *SIGKDD*, 2007, pp. 133–142.
[11] IBM, "IBM manifesto for Autonomic Computing: http://www.research.ibm.com/autonomic/." 2001.

[12] I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K. Hernandez, "Adaptive diagnosis in distributed systems," *IEEE Transactions on Neural Networks (special issue on Adaptive Learning Systems in Communication Networks)*, vol. 16, pp. 1088–1109, 2005.

[13] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, pp. 41–50, 2003.

[14] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, 2007.

[15] X. Zhang, C. Furtlehner, and M. Sebag, "Data streaming with affinity propagation," in *ECML/PKDD*, 2008, pp. 628–643.

[16] X. Zhang, C. Furtlehner, J. Perez, C. Germain-Renaud, and M. Sebag, "Toward autonomic grids: Analyzing the job flow with affinity streaming," in *SIGKDD*, 2009.

[17] E. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, pp. 100–115, 1954.

[18] D. Hinkley, "Inference about the change-point from cumulative sum tests," *Biometrika*, vol. 58, pp. 509–523, 1971.

[19] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: An application of large-scale online learning," in *ICML*, 2009, pp. 681–688.

[20] W. Fan, H. Wang, and P. S. Yu, "Active mining of data streams," in *SDM*, 2004.

[21] S. M. Muthukrishnan, "Data streams: Algorithms and applications," in *Found. Trends Theor. Comput. Sci.*, vol. 1. Now Publishers Inc., 2005, pp. 117–236.

[22] S. Nittel, K. T. Leung, and A. Braverman, "Scaling clustering algorithms for massive data sets using data streams," in *ICDE*, 2004, p. 830.

[23] P. S. Bradley, U. M. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," in *Knowledge Discovery and Data Mining*, 1998, pp. 9–15.

[24] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in *SIGMOD*, 1996, pp. 103–114.

[25] G. Dong, J. Han, L. V. S. Lakshmanan, J. Pei, H. Wang, and P. S. Yu, "Online mining of changes from data streams: Research problems and preliminary results," in *ACM SIGMOD Workshop on Management and Processing of Data Streams*, 2003.

[26] B.-R. Dai, J.-W. Huang, M.-Y. Yeh, and M.-S. Chen, "Adaptive clustering for multiple evolving streams," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 18, no. 9, pp. 1166–1180, Sep. 2006.

[27] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *SIGKDD*, 1996, pp. 226–231.

[28] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 23, pp. 859–874, 2011.

[29] Z. Harchaoui, "Kernel methods for detection (méthodes à noyaux pour la détection)," Ph.D. dissertation, TELECOM ParisTech, Paris, France, 2008.

[30] Z. Harchaoui, F. Bach, and E. Moulines, "Kernel change-point analysis," in *NIPS*, 2009, pp. 609–616.

[31] KDD'99, "KDD Cup 1999 data (computer network intrusion detection): http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html."

[32] W. Lee, S. J. Stolfo, and K. W. Mok, "A data mining framework for building intrusion detection models," in *Proceedings of the IEEE Symposium on Security and Privacy*, 1999, pp. 120–132.

[33] W. Wang, X. Guan, and X. Zhang, "Processing of massive audit data streams for real-time anomaly intrusion detection," *Computer Communications*, vol. 31, no. 1, pp. 58–72, 2008.

[34] S. Papadimitriou, A. Brockwell, and C. Faloutsos, "Adaptive, hands-off stream mining," in *VLDB*, 2003, pp. 560–571.

[35] A. Arasu and G. S. Manku, "Approximate counts and quantiles over sliding windows," in *PODS*, 2004, pp. 286–296.

[36] B. Babcock and C. Olston, "Distributed TopK monitoring," in *SIGMOD*, 2003, pp. 28–39.

[37] X. H. Dang, W.-K. Ng, and K.-L. Ong, "An error bound guarantee algorithm for online mining frequent sets over data streams," *Journal of Knowledge and Information Systems*, 2007.

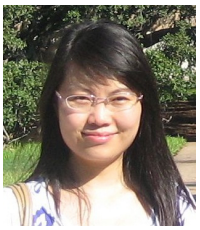
[38] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high speed data streams," in *SIGMOD*, 2003, pp. 523–528.

[39] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, "Finding hierarchical heavy hitters in streaming data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 4, 2008.

[40] M. Leone, Sumedha, and M. Weigt, "Clustering by soft-constraint affinity propagation: Applications to gene-expression data," *Bioinformatics*, vol. 23, p. 2708, 2007.

[41] X. Zhang, W. Wang, K. Norvag, and M. Sebag, "K-AP: Generating Specified K Clusters by Efficient Affinity Propagation," in *ICDM*, 2010, pp. 1187–1192.

[42] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual Symposium on Computational Geometry (SCG)*, 2004, pp. 253–262.



Xiangliang Zhang is currently Assistant Professor and directs the Machine Intelligence and kNowledge Engineering (MINE) Laboratory in the Division of Computer, Electrical and Mathematical Sciences and Engineering, King Abdullah University of Science and Technology (KAUST), Saudi Arabia. She was an European ERCIM research fellow in Norwegian University of Science and Technology, Norway, in 2010. She earned her Ph.D. degree in computer science from INRIA-Université Paris-Sud, France, in July 2010. She has authored or co-authored over 40 refereed papers in various journals and conferences. Her main research interests and experiences are in machine learning, data mining, and cloud computing.

Cyril Furtlehner is Researcher in INRIA-France since 2007. He graduated at Ecole Normale Supérieure in Paris, and obtained a PhD in Theoretical Physics at Université Paris 6 in 1997. His current research interests include message passing algorithms, discrete stochastic systems, and traffic modeling. He is PI of the ANR project Travesti (2009-2012), dealing with the inference of road traffic.

Cécile Germain-Renaud is a full Professor at Université Paris-Sud, where she earned her PhD in computer science in 1989. She servers on the Program Committee for main conferences like CCGrid and ICAC, and was the co-editor of the special issue of Journal of Grid Computing. Her research interests are in models for High Performance Computing, and in Autonomic Grid and Clouds. She is involved with the European Grid Initiative, and leads the Grid Observatory project.

Michèle Sebag holds a highly competitive Senior Researcher position at CNRS since 2002, and directs the TAO group since 2001. She graduated at Ecole Normale Supérieure in Maths, and received her PhD in computer science from Université Paris 6. She is ECCAI Fellow since 2011, member of the European steering committee of Machine Learning and Data Mining since 2010, program co-chair of the ECML/PKDD in 2010, on the editorial board of Machine Learning and Genetic Programming and Evolvable Hardware, and area chair or PC member of most international conferences in Machine Learning and Evolutionary Computation. She authored and co-authored more than 120 refereed papers in major international conferences and journals in Machine Learning and Evolutionary Computation.